UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/862,412	05/21/2001	Seth M. Demsey	MS160304.1/40062.100US01	8482

7590 08/11/2005

Homer L. Knearl
Merchant & Gould P.C.
P.O. Box 2903
Minneapolis, MN 55402-0903

EXAMINER	
BULLOCK JR, LEWIS ALEXANDER	
ART UNIT	**PAPER NUMBER**
2195	

DATE MAILED: 08/11/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/862,412	DEMSEY ET AL.	
	Examiner	Art Unit	
	Lewis A. Bullock, Jr.	2195	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *RCE filed 5/2/05*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-4,6-22,24,26,27,29-37,39,40 and 42-49* is/are pending in the application.

4a) Of the above claim(s) *42-49* is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-4,6-22,24,26,27,29-37,39 and 40* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *21 May 2001* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☒ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

Election/Restrictions

1. Restriction to one of the following inventions is required under 35 U.S.C. 121:

   I. Claims 1-4, 6-22, 24, 26, 27, 2937, 39 and 40 are, drawn to creating and sending of a customized library for executing an application, classified in class 719, subclass 331.

   II. Claims 42-49 are, drawn to monitoring resource usage to de-allocate least recently used applications/libraries when the resources are depleted, classified in class 718, subclass 104.

The inventions are distinct, each from the other because of the following reasons:

2. Inventions Group I and Group II are related as subcombinations disclosed as usable together in a single combination. The subcombinations are distinct from each other if they are shown to be separately usable. In the instant case, invention Group II has separate utility such as monitoring the least recently used applications and deflating them according to the resource limits. See MPEP § 806.05(d).

3. Because these inventions are distinct for the reasons given above and have acquired a separate status in the art as shown by their different classification, different search, and recognized divergent subject matter, restriction for examination purposes as indicated is proper.

4. Newly submitted claims 42-49 are directed to an invention that is independent or distinct from the invention originally claimed for the reasons given above.

Since applicant has received an action on the merits for the originally presented invention, this invention has been constructively elected by original presentation for prosecution on the merits. Accordingly, claims 42-49 are withdrawn from consideration as being directed to a non-elected invention. See 37 CFR 1.142(b) and MPEP § 821.03.

Claim Rejections - 35 USC § 101

5. 35 U.S.C. 101 reads as follows:

> Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

Claims 1-4, 6-22, 34-37, 39, and 40 are rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter. The cited claims detail a computer program product which as defined on page 4, lines 3-5 is a computer data signal. A data signal is not a tangible structure as proper under M.P.E.P. 2106. Therefore, the cited claims are rejection as being directed to non-statutory subject matter..

Claim Rejections - 35 USC § 112

6. Claims 26, 27 and 29-33 are rejected under 35 U.S.C. 112, second paragraph, as being incomplete for omitting essential structural cooperative relationships of elements, such omission amounting to a gap between the necessary structural connections. See MPEP § 2172.01. The omitted structural cooperative relationships are: the use of the catalog and filter module to generate a client composite list to the

claimed receipt module separately receiving the customized library. There should be some structural relationship that the composite list is used to retrieve the client-needed types that are stored in the customized library to be received by the receipt module. As currently stated, there is a gap between the operations of the filter module to the client module. The customized library appears to have no relation to the other components and their operations.

Claim Rejections - 35 USC § 103

7. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

8. Claims 34, 35 and 36 are rejected under 35 U.S.C. 103(a) as being unpatentable over BURRIDGE (U.S. Patent 6,918,106).

As to claim 34, BURRIDGE teaches a computer program product encoding a computer program for executing on a computer system (object oriented computer system), a computer process for creating a customized library (library file / JAR file), the computer process comprising: accessing a class library store (source having needed class / referenced library file); identifying one or more needed classes (needed application program units) referenced by the application (main program unit / application) but not included with the application; creating a library (new library file) (col. 7, lines 49-51); extracting the one or more needed classes from the class library store

(via placing a copy of the needed class in the library file); adding to the library metadata (JAR file information / manifest file / fields) describing the method, the metadata being extracted from the class library store in the extracting operation (via placing a copy of the needed class in a library file wherein the library file is a JAR file); and adding the one or more needed classes to the empty library to generate the customized library (via placing a copy of the needed class and its information in a library JAR file) (col. 9, line 53 – col. 10, line 27; col. 8, lines 54-63; col. 5, lines 16-45; col. 5, line 56 – 65; col. 5, line 66 – col. 6, line 10; col. 6, lines 15-28; col. 6, lines 50-52; col. 9, lines 3-13; col. 9, lines 23-29; abstract). However, BURRIDGE does not teach that the library is empty. It would be obvious to one skilled in the art at the time of the invention that since the library is created and then application files are placed in it, that the library file is initially empty and therefore would be obvious in view of the teachings of BURRIDGE that the library file is empty before metadata is added to the file.

As to claim 35, BURRIDGE teaches receiving a composite list specifying the needed classes (via receiving a JAR file of needed classes not loaded on the system as determined by the loader and stored in a library file) (col. 5, lines 16-45; col. 5, line 56 – 65; col. 5, line 66 – col. 6, line 10; col. 6, lines 15-28; col. 6, lines 50-52; col. 9, lines 3-13; col. 9, lines 23-29; abstract).

As to claim 36, BURRIDGE teaches adding one or more global data fields (fields) of each needed class to the library, the global data fields being extracted from the class

library store in the extracting operation (field for storing a class file) (col. 10, lines 19-28).

9. Claims 1-3, 6, 9, 11-20, 24, 26, 27, 29, 31, 37, 39 and 40 are rejected under 35 U.S.C. 103(a) as being unpatentable over BURRIDGE (U.S. Patent 6,918,106) in view of "Enterprise JavaBeans" by Monson-Haefel (HAEFEL).

As to claim 1, BURRIDGE teaches a computer program product encoding a computer program for executing on a computer system (object oriented computer system), a computer process for generating a customized library (library file / JAR file) for execution of an application (application) by a client (user system), the client including one or more client loaded types (program units stored in an initial application library file), the computer process comprising: identifying one or more application-referenced types (application program units) on which the application (main program unit) depends for execution but which are not part of the application (via the offline loader / runtime loader determining application program units that are loaded as determined by referenced library file); identifying one or more client needed types (needed application program units) required by the client to execute the application, based on the application referenced types and the client loaded types (via the loader determining at a subsequent invocation that the application file is not already loaded in the referenced library file and creating a new library file having the needed application file); generating the customized library, including the one or more client needed types (loader creating library file containing needed application file that is not loaded); and separately sending

the customized library (via loading the file when it is found or when all determined referenced files are placed in the library file) (col. 5, lines 16-45; col. 5, line 56 – 65; col. 5, line 66 – col. 6, line 10; col. 6, lines 15-28; col. 6, lines 50-52; col. 9, lines 3-13; col. 9, lines 23-29; abstract). It would be obvious to one skilled in the art at the time of the invention that the application is sent separate from the file because the application is executed in order to determine which program files to load (see also col. 9, lines 3-13). However, BURRIDGE does not teach explicitly teach that third parties develop the referenced types.

HAEFEL teaches the loading of referenced types (Java classes) in a library (JAR file) for an application wherein third parties develop the referenced types (pg. 30, "JAR files are ZIP files that are used specifically for packaging Java classes that are ready to be used in some type of application."; pg. 90, "As a packaging mechanism, however, the JAR file format is a very convenient way to "shrink-wrap" components and other software for delivery to third parties."). Therefore, it would be obvious to one skilled in the art at the time of the invention to combine the teachings of BURRIDGE with the teachings of HAEFEL in order to facilitate package classes to be used by some type of application (pg. 30).

As to claim 24, BURRIDGE teaches a customized library management system (object oriented computer system) for managing a customized class library (library file / JAR file), the system comprising: an application server module (source) receiving an application request and transmitting a requested application to a client (via the

application needing an application file that isn't loaded and loading it from a determined path/source); and a customized library generator creating a customized library (library file / JAR file) (via loader creating library file containing needed application file that is not loaded) and separately sending the customized library (library file / JAR file) to the client (via loading the file when it is found or when all determined referenced files are placed in the library file), the customized library including one or more client needed types, which are application referenced types not loaded on the client but required by the application to execute (loader creating library file containing needed application file that is not loaded) (col. 5, lines 16-45; col. 5, line 56 – 65; col. 5, line 66 – col. 6, line 10; col. 6, lines 15-28; col. 6, lines 50-52; col. 9, lines 3-13; col. 9, lines 23-29; abstract). It would be obvious to one skilled in the art at the time of the invention that the application is sent separate from the file because the application is executed in order to determine which program files to load (see also col. 9, lines 3-13). However, BURRIDGE does not explicitly teach that third parties develop the referenced types.

HAEFEL teaches the loading of referenced types (Java classes) in a library (JAR file) for an application wherein third parties develop the referenced types (pg. 30, "JAR files are ZIP files that are used specifically for packaging Java classes that are ready to be used in some type of application."; pg. 90, "As a packaging mechanism, however, the JAR file format is a very convenient way to "shrink-wrap" components and other software for delivery to third parties."). Therefore, it would be obvious to one skilled in the art at the time of the invention to combine the teachings of BURRIDGE with the

teachings of HAEFEL in order to facilitate package classes to be used by some type of application (pg. 30).

As to claims 26, BURRIDGE teaches a customized library management system (object oriented computer system) for separately receiving a customized library (library file / JAR file) associated with an application (main program unit / application), the system comprising: a catalog (referenced application library file) identifying one or more client loaded types that are loaded in a client (via program units stored in an initial application library file referenced by the main program unit); a filter module comparing the catalog to a list of one or more application referenced types, which are required by the application to execute (via the loader determining at a subsequent invocation that the application file is not already loaded in the referenced library file and creating a new library file having the needed application file), and generating a client composite list (another application library file) to identify one or more client needed types (loader creating library file containing needed application file that is not loaded); and a client receipt module for separately receiving the customized library and the application (via loading the file when it is found or when all determined referenced files are placed in the library file) (col. 5, lines 16-45; col. 5, line 56 – 65; col. 5, line 66 – col. 6, line 10; col. 6, lines 15-28; col. 6, lines 50-52; col. 9, lines 3-13; col. 9, lines 23-29; abstract). It would be obvious to one skilled in the art at the time of the invention that the application is sent separate from the file because the application is executed in order to determine which

program files to load (see also col. 9, lines 3-13). However, BURRIDGE does not teach explicitly teach that third parties develop the referenced types.

HAEFEL teaches the loading of referenced types (Java classes) in a library (JAR file) for an application wherein third parties develop the referenced types (pg. 30, "JAR files are ZIP files that are used specifically for packaging Java classes that are ready to be used in some type of application."; pg. 90, "As a packaging mechanism, however, the JAR file format is a very convenient way to "shrink-wrap" components and other software for delivery to third parties."). Therefore, it would be obvious to one skilled in the art at the time of the invention to combine the teachings of BURRIDGE with the teachings of HAEFEL in order to facilitate package classes to be used by some type of application (pg. 30).

As to claim 2, BURRIDGE teaches inserting only client-needed types into the customized library (col. 5, lines 16-46).

As to claim 3, BURRIDGE teaches inserting all client-needed types and all client-loaded types into the customized library (col. 5, lines 16-46).

As to claim 6, BURRIDGE teaches excluding from the customized library one or more non-identified types, each non-identified type not being referenced by the application (via the developer removing the reference) (col. 7, line 59 – col. 8, line 2).

As to claim 9, BURRIDGE teaches identifying the one or more client-loaded types (program units stored in an initial application library file); and generating a client composite list to identify the one or more client-needed types, the one or more client needed types including the one or more application-referenced types not loaded on the client (via the loader determining at a subsequent invocation that the application file is not already loaded in the referenced library file and creating a new library file having the needed application file) (col. 5, lines 16-45; col. 5, line 56 – 65; col. 5, line 66 – col. 6, line 10; col. 6, lines 15-28; col. 6, lines 50-52; col. 9, lines 3-13; col. 9, lines 23-29; abstract).

As to claim 11, BURRIDGE teaches accessing a client catalog specifying the one or more client loaded types loaded on the client (program units stored in an initial application library file); and examining the client catalog to identify the one or more client loaded types (col. 5, lines 16-45; col. 5, line 56 – 65; col. 5, line 66 – col. 6, line 10; col. 6, lines 15-28; col. 6, lines 50-52; col. 9, lines 3-13; col. 9, lines 23-29; abstract).

As to claim 12, BURRIDGE teaches identifying the one or more client loaded types by receiving a list of the client loaded types from the client (via a loader using a reference to the initial application library file to search for a invoked application file); and evaluating the client loaded types against the application referenced types to identify the client needed types (col. 5, lines 16-45; col. 5, line 56 – 65; col. 5, line 66 – col. 6, line 10; col. 6, lines 15-28; col. 6, lines 50-52; col. 9, lines 3-13; col. 9, lines 23-29; abstract).

As to claim 13, BURRIDGE teaches receiving a client composite list specifying the one or more application-referenced types not loaded on the client (via receiving a JAR file of needed classes not loaded on the system as determined by the loader and stored in a library file) (col. 5, lines 16-45; col. 5, line 56 – 65; col. 5, line 66 – col. 6, line 10; col. 6, lines 15-28; col. 6, lines 50-52; col. 9, lines 3-13; col. 9, lines 23-29; abstract).

As to claim 14, BURRIDGE teaches creating a new library (col. 7, lines 49-51); and adding each of the one or more client needed types to the new library to provide the customized library (via creating a new library and adding the needed classes to the library) (col. 5, lines 16-45; col. 5, line 56 – 65; col. 5, line 66 – col. 6, line 10; col. 6, lines 15-28; col. 6, lines 50-52; col. 9, lines 3-13; col. 9, lines 23-29; abstract). However, BURRIDGE does not teach that the library is empty, i.e. having no types. It would be obvious to one skilled in the art at the time of the invention that since the library is created and then application files are placed in it, that the library file is initially empty and therefore would be obvious in view of the teachings of BURRIDGE that the library file is empty before metadata is added to the file.

As to claim 15, BURRIDGE teaches adding one or more global data fields of each client needed type to the new library (field for storing a class file) (col. 10, lines 19-28).

As to claims 16 and 17, HAEFEL teaches the loading of referenced types (Java classes) in a library (JAR file) for an application wherein third parties develop the referenced types (pg. 30, "JAR files are ZIP files that are used specifically for packaging Java classes that are ready to be used in some type of application."; pg. 90, "As a packaging mechanism, however, the JAR file format is a very convenient way to "shrink-wrap" components and other software for delivery to third parties."). Official Notice is taken in that it is well known in the art that Java classes have methods and data fields and therefore it would be obvious that since referenced types are packaged and they are classes, that the information would include method signatures and code regarding the method for package.

As to claims 18-20, BURRIDGE an application-referenced type or client-loaded type includes a class (col. 8, lines 54-67; col. 9, lines 3-16).

As to claim 27, BURRIDGE teaches a client request module requesting the application from a server (via the loader loading the main program unit) (col. 1, lines 28-29; col. 5, lines 56-59).

As to claim 29, BURRIDGE teaches identifying the application to be optimized (col. 5, lines 56-59). It is obvious to one skilled in the art at the time of the invention that in order to identify the application, an identifier would have to be associated with the application.

As to claim 31, BURRIDGE teaches an install point indicator associated with the application (pathnames for application files) (col. 5, lines 55-65).

As to claims 37, 39 and 40, BURRIDGE substantially disclose the invention above. However, BURRIDGE does not teach the cited functionality. HAEFEL teaches the loading of referenced types (Java classes) in a library (JAR file) for an application wherein third parties develop the referenced types (pg. 30, "JAR files are ZIP files that are used specifically for packaging Java classes that are ready to be used in some type of application."; pg. 90, "As a packaging mechanism, however, the JAR file format is a very convenient way to "shrink-wrap" components and other software for delivery to third parties."). Official Notice is taken in that it is well known in the art that Java classes have methods and data fields and therefore it would be obvious that since referenced types are packaged and they are classes, that the information would include method signatures and code regarding the method for package. Therefore, it would be obvious to one skilled in the art at the time of the invention to combine the teachings of BURRIDGE with the teachings of HAEFEL in order to facilitate package classes to be used by some type of application (pg. 30).

10. Claims 4 and 10 are rejected under 35 U.S.C. 103(a) as being unpatentable over BURRIDGE in view of HAEFEL as applied to claim 1 above, and further in view of

"Method to Update Java Class Library in Client Computer at Runtime" by IBM Technical Disclosure.

As to claim 4, the cited combination substantially teaches the claims as detailed above. However, the cited combination does not teach using a version identifier. IBM teaches identifying a version identifier of loaded classes such that the class loader checks the version of the request class in the client and the server before loading the class (pgs 1-2). Therefore, it would be obvious to combine the teachings of BURRIDGE with HAEFEL and IBM in order to improve the efficiency of the version control of Java classes distributed in the client computers (pg. 1).

As to claim 10, the cited combination substantially teaches the claims as detailed above. However, the cited combination does not teach using a version identifier. IBM teaches identifying a version identifier of loaded classes such that the class loader checks the version of the request class in the client and the server before loading the class such that new versions of requested classes are downloaded as determined (pgs 1-2). Therefore, it would be obvious to combine the teachings of BURRIDGE with HAEFEL and IBM in order to improve the efficiency of the version control of Java clases distributed in the client computers (pg. 1).

11. Claims 7, 8, 30, 32 and 33 are rejected under 35 U.S.C. 103(a) as being unpatentable over BURRIDGE in view of HAEFEL as applied to claims 1 and 26 above, and further in view of CHAN (U.S. Patent 6,470,494).

As to claims 7 and 8, BURRIDGE and HAEFEL substantially disclose the invention above. However, neither of the references teaches the generating a dependency list. CHAN teaches recursively examining the application to identify one or more type references (classes determined to be needed) associated with an application referenced type (via upon the execution of an application, a particular class needs to be loaded) (col. 5, lines 65 – col. 6, line 35; and generating a dependency list identifying the one or more application referenced types based on the type references (via analyzing a classes loaded in the jar file to determine if other classes are needed and adding those classes o the input jar file) (col. 7, line 51 – col. 8, line 32). Therefore, it would be obvious to one skilled in the art at the time of the invention to combine the teachings of BURRIDGE with the teachings of HAEFEL and CHAN in order to facilitate the loading of classes without having to specify the location of the classes prior to run-time (col. 3, line 66 – col. 4, line 1).

As to claims 30, 32 and 33, BURRIDGE and HAEFEL substantially disclose the invention above. However, neither of the references teaches the generating a dependency information. CHAN teaches recursively examining the application to identify one or more type references (classes determined to be needed) associated with an application referenced type (via upon the execution of an application, a particular class needs to be loaded) (col. 5, lines 65 – col. 6, line 35; and generating a dependency list identifying the one or more application referenced types based on the type references (via analyzing a classes loaded in the jar file to determine if other

classes are needed and adding those classes o the input jar file) (col. 7, line 51 – col. 8, line 32). Therefore, it would be obvious to one skilled in the art at the time of the invention to combine the teachings of BURRIDGE with the teachings of HAEFEL and CHAN in order to facilitate the loading of classes without having to specify the location of the classes prior to run-time (col. 3, line 66 – col. 4, line 1)

12. Claims 21 and 22 are rejected under 35 U.S.C. 103(a) as being unpatentable over BURRIDGE in view of HAEFEL as applied to claim 1 above, and further in view of MENACHEMI (U.S. Patent Application Publication 2002/0103810 A1).

As to claims 21 and 22, BURRIDGE and HAEFEL substantially disclose the invention above. However, neither reference teach the use of a device profile. MENACHEMI teaches downloading of a class and a specification of an application, i.e. device profile, by a class loader by creating an empty class and loading the class and its class details in the empty class (pg. 5, paragraphs 0066 – 0070) and receiving a device profile (device profile and specification) specifying a characteristic of the client (pg. 4, lines 0054-0056); and excluding an attribute from the customized library (class), if the attribute is incompatible with the characteristic of the client (via executing part or all of the application based on the device profile and specification received) (pg. 4, lines 0057). Therefore, it would be obvious to one skilled in the art at the time of the invention to combine the teachings of BURRIDGE with the teachings of HAEFEL and MENACHEMI in order to dynamically building at least part of an application at run-time (pg. 2, paragraph 0038).

Conclusion

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Lewis A. Bullock, Jr. whose telephone number is (571) 272-3759. The examiner can normally be reached on Monday-Friday, 8:30 - 5:00.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Meng An can be reached on (571) 272-3756. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

August 4, 2005

LEWIS A. BULLOCK, JR.
PRIMARY EXAMINER